ILLINOIS INSTITUTE OF TECHNOLOGY

CS520 Data Integration, Warehousing, and Provenance

3. Schema Matching and Mapping

IIT DBGroup

Boris Glavic

http://www.cs.iit.edu/~glavic/

http://www.cs.iit.edu/~cs520/

http://www.cs.iit.edu/~dbgroup/

0

Outline

0) Course Info
1) Introduction
2) Data Preparation and Cleaning
3) **Schema matching and mapping**
4) Virtual Data Integration
5) Data Exchange
6) Data Warehousing
7) Big Data Analytics
8) Data Provenance

1

3. Why matching and mapping?

- **Problem: Schema Heterogeneity**
  - Sources with different schemas store overlapping information
  - Want to be able to translate data from one schema into a different schema
    - Datawarehousing
    - Data exchange
  - Want to be able to translate queries against one schema into queries against another schema
    - Virtual dataintegration

2

3. Why matching and mapping?

- **Problem: Schema Heterogeneity**
  - We need to know how elements of different schemas are related!
  - **Schema matching**
    - Simple relationships **such as attribute name of relation person in the one schema corresponds to attribute lastname of relation employee in the other schema**
  - **Schema mapping**
    - Also model correlations and missing information such as links caused by foreign key constraints

3

3. Why matching and mapping?

- **Why both mapping and matching**
  - Split complex problem into simpler subproblems
    - Determine matches and then correlate with constraint information into mappings
  - Some tasks only require matches
    - E.g., matches can be used to determine attributes storing the same information in data fusion
  - Mappings are a natural generalization of matchings

4

3. Overview

- Topics covered in this part
  - **Schema Matching**
  - Schema Mappings and Mapping Languages

5

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Problem: Schema Matching**
  - Given two (or more schemas)
    - For now called **source** and **target**
  - Determine how elements are related
    - Attributes are representing the same information
      - **name = lastname**
    - Attribute can be translated into an attribute
      - **MonthlySalary * 12 = Yearly Salary**
    - **1-1** matches vs. **M-N** matches
      - **name** to **lastname**
      - **name** to **concat(firstname, lastname)**

6

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Why is this hard?**
  - **Insufficient information**: schema does not capture full semantics of a domain
  - **Schemas can be misleading**:
    - E.g, attributes are not necessarily descriptive
    - E.g., finding the right way to translate attributes not obvious

7

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

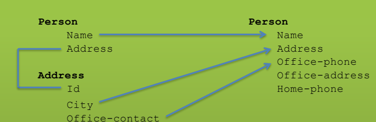
- **What information to consider?**
  - Attribute names
    - or more generally element names
  - Structure
    - e.g., belonging to the same relation
  - Data
    - Not always available
- **Need to consider multiple types to get reasonable matching quality**
  - Single types of information not predictable enough

8

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

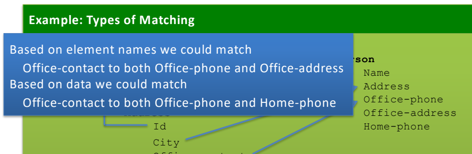
Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Chicago, IL 60655	(333) 323 3344
Alice	Chicago	(312) 555 7777	Chicago, IL 60633	(123) 323 3344
Bob	New York	(465) 123 1234	New York, NY 55443	(888) 323 3344

9

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

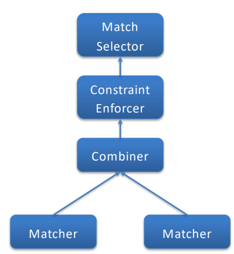
Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343	Chicago, IL 60655	(333) 323 3344
Alice	Chicago	(312) 555 7777	Chicago, IL 60633	(123) 323 3344
Bob	New York	(465) 123 1234	New York, NY 55443	(888) 323 3344

10

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Typical Matching System Architecture**



Determine actual matches

Use constraints to modify similarity matrix

Combine individual similarity matrices

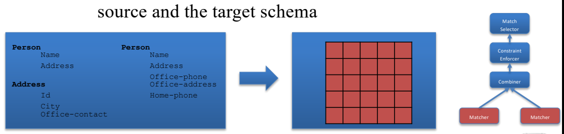
Each matcher uses one type of information to compute similarity matrix

11

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Matcher**
  - **Input:** Schemas
    - Maybe also data, documentation
  - **Output**: Similarity matrix
    - Storing value [0,1] for each pair of elements from the source and the target schema



12

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

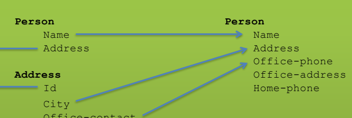
- **Name-Based Matchers**
  - String similarities measures
    - E.g., Jaccard and other measure we have discussed
  - Preprocessing
    - Tokenization?
    - Normalization
      - Expand abbreviations and replace synonyms
    - Remove stop words
      - In, and, the

13

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching



	Name	Address	Office-phone	Office-address	Home-phone
Name	1	0	0	0	0
Address	0	1	0	0.4	0
Id	0	0	0	0	0
City	0	0	0	0	0
Office-contact	0	0	0.5	0.5	0

14

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Data-Based Matchers**
  - Determine how similar the values of two attributes are
  - Some techniques
    - Recognizers
      - Dictionaries, regular expressions, rules
    - Overlap matcher
      - Compute overlap of values in the two attributes
    - Classifiers

15

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Recognizers**
  - Dictionaries
    - Countries, states, person names
  - Regular expression matchers
    - **Phone numbers:** `(\+\d{2})? \(\d{3}\) \d{3} \d{4}`

16

3.1 Schema Matching

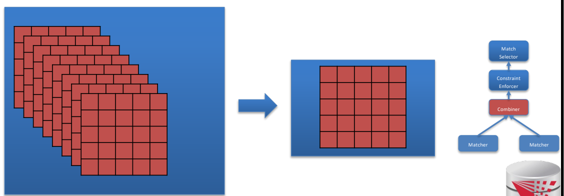
ILLINOIS INSTITUTE OF TECHNOLOGY

- **Overlap of attribute domains**
  - Each attribute value is a token
  - Use set-based similarity measure such as Jaccard
- **Classifier**
  - Train classifier to identify values of one attribute **A** from the source
    - Training set are values from **A** as positive examples and values of other attributes as negative examples
  - Apply classifier to all values of attributes from target schema
    - Aggregate into similarity score

17

3.1 Schema Matching

- **Combiner**
  - **Input:** Similarity matrices
    - Output of the individual matchers
  - **Output**: Single Similarity matrix



18

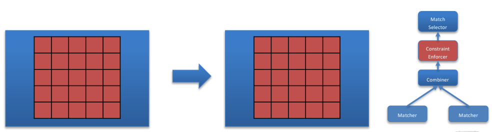
3.1 Schema Matching

- **Combiner**
  - Merge similarity matrices produced by the matchers into single matrix
  - Typical strategies
    - Average, Minimum, Max
    - Weighted combinations
    - Some script

19

3.1 Schema Matching

- **Constraint Enforcer**
  - **Input:** Similarity matrix
    - Output of Combiner
  - **Output**: Similarity matrix



20

3.1 Schema Matching

- **Constraint Enforcer**
  - Determine most probably match by assigning each attribute from source to one target attribute
    - Multiple similarity scores to get likelihood of match combination to be true
  - Encode domain knowledge into constraints
    - **Hard constraints**: Only consider match combinations that fulfill constraints
    - **Soft constraints**: violating constraints results in penalty of scores
      - Assign cost for each constraint
  - Return combination that has the maximal score

21

3.1 Schema Matching

Example: Constraints

Constraint 1: An attribute matched to **source.cust-phone** has to get a score of 1 from the phone regexpr matcher

Constraint 2: Any attribute matched to **source.fax** has to have fax in its name

Constraint 3: If an attribute is matched to **source.firstname** with score > 0.9 then there has to be another attribute from the same target table that is matched to **source.lastname** with score > 0.9

22

3.1 Schema Matching

- **How to search match combinations**
  - Full search
    - Exponentially many combinations potentially
  - Informed search approaches
    - A* search
  - Local propagation
    - Only local optimizations

23

3.1 Schema Matching

- **A* search**
  - Given a search problem
    - Set of states: start state, goal states
    - Transitions about states
    - Costs associated with transitions
    - Find cheapest path from start to goal states
  - Need admissible heuristics **h**
    - For a path **p**, **h** computes lower bound for any path from start to goal with prefix **p**
  - Backtracking best-first search
    - Choose next state with lowest estimated cost
    - Expand it in all possible ways

3.1 Schema Matching

- **A* search**
  - Estimated cost of a state **f(n) = g(n) + h(n)**
    - **g(n)** = cost of path from start state to **n**
    - **h(n)** = lower bound for path from **n** to goal state
  - No path reaching the goal state from **n** can have a total cost lower than **f(n)**

3.1 Schema Matching

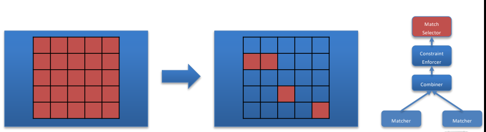
- **Algorithm**
  - Data structures
    - Keep a priority queue **q** of states sorted on f(n)
      - Initialize with start state
    - Keep set **v** of already visited nodes
      - Initially empty
  - While **q** is not empty
    - pop state **s** from head of **q**
    - If **s** is goal state return
    - Foreach **s'** that is direct neighbor of **s**
      - If **s'** not in v
      - Compute **f(s')** and insert **s'** into **q**

3.1 Schema Matching

- **Application to constraint enforcing**
  - Source attributes: A_1 to A_n
  - Target attributes: B_1 to B_m
  - States
    - Vector of length n with values B_i or * indicating that no choice has not been taken
    - [B_1, *, *, B_3]
  - Initial state
    - [*, *, *, *]
  - Goal states
    - All states without *

3.1 Schema Matching

- **Match Selector**
  - **Input:** Similarity matrix
    - Output of the individual matchers
  - **Output**: Matches



3.1 Schema Matching

- **Match Selection**
  - Merge similarity matrices produced by the matchers into single matrix
  - Typical strategies
    - Average, Minimum, Max
    - Weighted combinations
    - Some script

3.1 Schema Matching

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Many-to-many matchers**
  - Combine multiple columns using a set of functions
    - **E.g., concat, +, currency exchange, unit exchange**
  - Large or even unlimited search space
  - -> need method that explores interesting part of the search space
  - Specific searchers
    - Only concatenation of columns (limit number of combinations, e.g., 2)

30 CS520 - 3) Matching and Mapping

30

3. Overview

ILLINOIS INSTITUTE OF TECHNOLOGY

- Topics covered in this part
  - Schema Matching
  - **Schema Mappings and Mapping Languages**

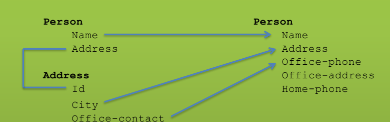
31 CS520 - 3) Matching and Mapping

31

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Matching Result



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Assume: We have data in the source as shown above

What data should we create in the target? Copy values based on matches?

32 CS520 - 3) Matching and Mapping

32

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

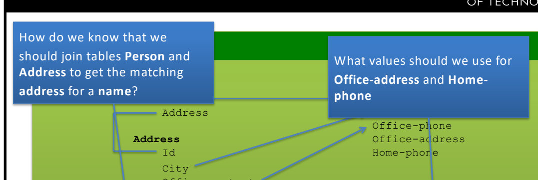
- Matches do not determine completely how to create the target instance data! (**Data Exchange**)
  - How do we choose values for attributes that do not have a match?
  - How do we combine data from different source tables?
- Matches do not determine completely what the answers to queries over a mediated schema should be! (**Virtual Data Integration**)

33 CS520 - 3) Matching and Mapping

33

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY



Name	Address
Peter	1
Alice	3
Bob	3

Id	City	Office-contact
1	Chicago	(312) 123 4343
2	Chicago	(312) 555 7777
3	New York	(465) 123 1234

Name	Address	Office-phone	Office-address	Home-phone
Peter	Chicago	(312) 123 4343		
Alice	Chicago	(312) 555 7777		
Bob	New York	(465) 123 1234		

34 CS520 - 3) Matching and Mapping

34

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Schema mappings**
  - Generalize matches
  - Describe relationship between instances of schemas
  - Mapping languages
    - LAV, GAV, GLAV
    - Mapping as Dependencies: tuple-generating dependencies
- **Mapping generation**
  - **Input**: Matches, Schema constraints
  - **Output**: Schema mappings

35 CS520 - 3) Matching and Mapping

35

3.2 Schema Mapping

- **Instance-based definition of mappings**
  - Global schema **G**
  - Local schemas $\mathbf{S_1}$ to $\mathbf{S_n}$
  - Mapping **M** can be expressed as for each set of instances of the local schemas what are allowed instances of the global schema
    - Subset of $(\mathbf{I_G}$ x $\mathbf{I_1}$ x … x $\mathbf{I_n})$
  - Useful as a different way to think about mappings, but not a practical way to define mappings

3.2 Schema Mapping

- **Certain answers**
  - Given mapping **M** and **Q**
  - Instances $\mathbf{I_1}$ to $\mathbf{I_n}$ for $\mathbf{S_1}$ to $\mathbf{S_n}$
  - Tuple **t** is a certain answer for **Q** over $\mathbf{I_1}$ to $\mathbf{I_n}$
    - If for every instance $\mathbf{I_G}$ so that $(\mathbf{I_G}$ x $\mathbf{I_1}$ x … x $\mathbf{I_n})$ in **M** then **t** in $\mathbf{Q(I_G)}$

3.2 Schema Mapping

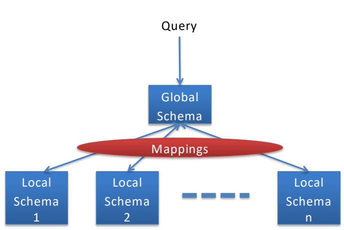
- **Languages for Specifying Mappings**
- **Describing mappings as inclusion relationships between views:**
  - Global as View (**GAV**)
  - Local as View (**LAV**)
  - Global and Local as View (**GLAV**)
- **Describing mappings as dependencies**
  - Source-to-target tuple-generating dependencies (**st-tgds**)

3.2 Schema Mapping

- **Describing mappings as inclusion relationships between views:**
  - Global as View (**GAV**)
  - Local as View (**LAV**)
  - Global and Local as View (**GLAV**)
- Terminology stems from virtual integration
  - Given a **global** (or mediated, or virtual) schema
  - A set of data sources (**local** schemas)
  - Compute answers to queries written against the global schema using the local data sources

3.2 Schema Mapping

- **Excursion Virtual Data Integration**
  - More in next section of the course



3.2 Schema Mapping

- **Global-as-view (GAV)**
  - Express the global schema as views over the local schemata
  - What query language do we support?
    - CQ, UCQ, SQL, …?
  - **Closed vs. open world** assumption
    - Closed world: $R = Q(S_1,\ldots,S_n)$
      - Content of global relation R is defined as the result of query Q over the sources
    - Open world: $R \supseteq Q(S_1,\ldots,S_n)$
      - Relation R has to contain the result of query Q, but may contain additional tuples

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: GAV

```
Local Schema                 Global Schema
Person                       Person
    Name                         Name
    Address                      Address
                                 Office-phone
Address
    Id
    City
    Office-contact
```

```
Q(X,Z,A) :- Person(X,Z,A)
= Q(X,Z,A) :- Person(X,Y), Address(Y,Z,A)
```

Since heads of LHS and RHS have to be the same we can use simpler notation without the head of the view expression:

```
Person(X,Z,A) = Person(X,Y), Address(Y,Z,A)
```

42 CS520 - 3) Matching and Mapping

42

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: GAV not possible

```
Local Schema                 Global Schema
Person                       Person
    Name                         Name
    Address                      Address
                                 Office-phone
Address                          Home-phone
    Id
    City
    Office-contact
```

```
Q(X',Y',Z',A') :- Person(X',Y',Z',A')
= Q(X,Z,A, ????) :- Person(X,Y), Address(Y,Z,A)
```

Cannot be expressed as GAV mapping! No way to compute the Home-phone attribute values since there is no correspondence with a source attribute!

43 CS520 - 3) Matching and Mapping

43

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-as-view (GAV)**
- **Solutions (mapping M)**
  - Unique data exchange solution (later)
  - Intuitively, execute queries over local instance that produced global instance

44 CS520 - 3) Matching and Mapping

44

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-as-view (GAV)**
- **Answering Queries**
  - Simply replace references to global tables with the view definition
- Mapping **R(X,Y) = S(X,Y), T(Y,Z)**
- **Q(X) :- R(X,Y)**
- Rewrite into
- **Q(X) :- S(X,Y), T(Y,Z)**

45 CS520 - 3) Matching and Mapping

45

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: GAV – query answering

```
Local Schema                 Global Schema
P1                           P2
    Name                         Name
    Address                      Address
                                 Office-phone
Address
    Id
    City
    Office-contact
```

GAV mapping:

```
P2(X,Z,A) = P1(X,Y), Address(Y,Z,A)
```

Query – Select Name from Persons

```
Q(A) :- P2(A,B,C)
```

View unfolding: Replace P2 with its definition

```
Q(A) :- P1(A,Y), Address(Y,B,C)
```

46 CS520 - 3) Matching and Mapping

46

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-as-view (GAV) Discussion**
  - Hard to add new source
    - -> have to rewrite the view definitions
  - Does not deal with missing values
  - Easy query processing
    - -> view unfolding

47 CS520 - 3) Matching and Mapping

47

3.2 Schema Mapping

- **Local-as-view (LAV)**
  - Express the local schema as views over the global schemata
  - What query language do we support?
    - CQ, UCQ, SQL, …?
  - **Closed vs. open world** assumption
    - Closed world: $S_{ij} = Q(G)$
      - Content of local relation S_{ij} is defined as the result of query Q over the sources
    - Open world: $S_{ij} \supseteq Q(G)$
      - Local relation S_{ij} has to contain the result of query Q, but may contain additional tuples

48 CS520 - 3) Matching and Mapping

48

3.2 Schema Mapping

Example: LAV

Local Schema		Global Schema
Person		**P2**
Name	→	Name
City	→	Address
Office-contact	→	Office-phone
		Office-address
		Home-phone

```
Person(X,Y,Z) = P2(X,Y,Z,A,B)
```

49 CS520 - 3) Matching and Mapping

49

3.2 Schema Mapping

Example: LAV not possible

Local Schema		Global Schema
Person		**Person**
Name	→ Name	Name
Address		Address
Address		Office-phone
Id		Office-address
City	→ Address	Home-phone
Office-contact	→ Office-phone	

Cannot deal with attributes from the local schema that do not have a correspondence with attributes in the global schema

```
Person(X,???) = Person(X,Y,Z,A,B)
Address(???,Y,Z) = Person(X,Y,Z,A,B)
```

50 CS520 - 3) Matching and Mapping

50

3.2 Schema Mapping

- **Local-as-view (LAV)**
- **Solutions (mapping M)**
  - Incompleteness possible
  - => There may exist many solutions

51 CS520 - 3) Matching and Mapping

51

3.2 Schema Mapping

- **Local-as-view (LAV)**
- **Answering Queries**
  - Need to find equivalent query using only the views (this is a hard problem, more in next course section)
- Mapping **S(X,Z) = R(X,Y), T(Y,Z)**
- **Q(X) :- R(X,Y)**
- Rewrite into ???
  - Need to come up with missing values
  - Give up query equivalence?

52 CS520 - 3) Matching and Mapping

52

3.2 Schema Mapping

- **Local-as-view (LAV) Discussion**
  - Easy to add new sources
    - -> have to write a new view definition
    - May take some time to get used to expressing sources like that
  - Still does not deal gracefully with all cases of missing values
    - Loosing correlation
  - Hard query processing
    - Equivalent rewriting using views only
    - Later: give up equivalence

53 CS520 - 3) Matching and Mapping

53

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Global-Local-as-view (GLAV)**
  - Express both sides of the constraint as queries
  - What query language do we support?
    - CQ, UCQ, SQL, …?
  - **Closed vs. open world** assumption
    - Closed world: Q'(G) = Q(S)
    - Open world: Q'(G) ⊇Q(S)

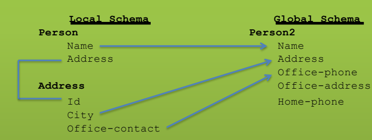
54 CS520 - 3) Matching and Mapping

54

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: GLAV




```
Source: Q(X,Y,Z) :- Person(X',Y'), Address(Y',Z',A')
=
Target: Q(X',Y',Z') :- Person2(X',Y',Z',A',B')
```

55 CS520 - 3) Matching and Mapping

55

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Local-as-view (GLAV) Discussion**
  - Kind of best of both worlds (almost)
  - Complexity of query answering is the same as for LAV
  - Can address the lost correlation and missing values problems we observed using GAV and LAV

56 CS520 - 3) Matching and Mapping

56

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Source-to-target tuple-generating dependencies (st-tgds)**
  - Logical way of expressing GLAV mappings
    - LHS formula is a conjunction of source (local) relation atoms (and comparisons
    - RHS formula is a conjunction of target (global) relation atoms and comparisons

$$\forall \vec{x} : \phi(\vec{x}) \rightarrow \exists \vec{y} : \psi(\vec{x}, \vec{y})$$

  - Equivalence to a containment constraint:

    Q'(G) ⊇Q(S)

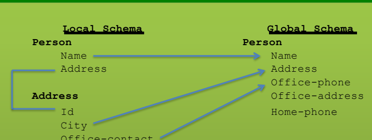
57 CS520 - 3) Matching and Mapping

57

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

Example: Types of Matching



$$\forall x, y, z, a : Person(x, y) \wedge Address(y, z, a) \rightarrow \exists b, c : Person(x, z, a, b, c)$$

```
Source: Q(X,Y,Z) :- Person(X',Y'), Address(Y',Z',A')
=
Target: Q(X',Y',Z') :- Person(X',Y',Z',A',B')
```

58 CS520 - 3) Matching and Mapping

58

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Generating Schema Mappings**
  - **Input**: Schemas (Constraints), matches
  - **Output**: Schema mappings
- Ideas:
  - Schema matches tell us which source attributes should be copied to which target attributes
  - Foreign key constraints tell us how to join in the source and target to not loose information

59 CS520 - 3) Matching and Mapping

59

3.2 Schema Mapping

- **Clio**
  - Clio is a data exchange system prototype developed by IBM and University of Toronto researchers
  - The concepts developed for Clio have been implemented in IBM InfoSphere Data Architect
  - Clio does matching, mapping generation, and data exchange
    - For now let us focus on the mapping generation

60

3.2 Schema Mapping

- **Clio Mapping Generation Algorithm**
  - **Inputs**: Source and Target schemas, matches
  - **Output**: Mapping from source to target schema
  - Note, Clio works for nested schemas such as XML too not just for relational data.
    - Here we will look at the relational model part only

61

3.2 Schema Mapping

- **Clio Algorithm Steps**
  - 1) Use **foreign keys** to determine all reasonable ways of **joining** data within the source and the target schema
    - Each alternative of joining tables in the source/target is called a logical association
  - 2) For each pair of **source-target logical associations**: Correlate this information with the matches to determine candidate mappings

62

3.2 Schema Mapping

- **Clio Algorithm: 1) Find logical associations**
  - This part relies on the **chase** procedure that first introduced to test implication of functional dependencies (‘77)
  - The idea is that we start use a representation of foreign keys are **inclusion dependencies** (tgds)
    - There are also chase procedures that consider **edgs** (e.g., PKs)
  - Starting point are all single relational atoms
    - E.g., R(X,Y)

63

3.2 Schema Mapping

- **Chase step**
  - Works on **tabelau**: set of relational atoms
  - A chase step takes one tgd t where the LHS is fulfilled and the RHS is not fulfilled
    - We fulfill the tgd t by adding new atoms to the tableau and mapping variables from t to the actually occuring variables from the current tablau
- **Chase**
  - Applying the chase until no more changes
  - Note: if there are cyclic constraints this may not terminate

64

3.2 Schema Mapping

- **Clio Algorithm: 1) Find logical associations**
  - Compute chase R(X) for each atom R in source and target
  - Each chase result is a logical association
  - Intuitively, each such logical association is a possible way to join relations in a schema based on the FK constraints

65

3.2 Schema Mapping

ILLINOIS INSTITUTE OF TECHNOLOGY

- **Clio Algorithm: 2) Generate Candidate Mappings**
  - For each pair of logical association $\mathbf{A_S}$ in the source and $\mathbf{A_T}$ in the target produced in step 1
  - Find the matches that are covered by $\mathbf{A_S}$ and $\mathbf{A_T}$
    - Matches that lead from an element of $\mathbf{A_S}$ to an element from $\mathbf{A_T}$
  - If there is at least one such match then create mapping by equating variables as indicated by the matches and create st-tgd with $\mathbf{A_S}$ in LHS and $\mathbf{A_T}$ in RHS

66 CS520 - 3) Matching and Mapping

66

Outline

ILLINOIS INSTITUTE OF TECHNOLOGY

0) Course Info
1) Introduction
2) Data Preparation and Cleaning
3) Schema matching and mapping
4) Virtual Data Integration
5) Data Exchange
6) Data Warehousing
7) Big Data Analytics
8) Data Provenance

67 CS520 - 3) Matching and Mapping

67